DOI https://doi.org/10.15407/csc.2022.04.013
УДК 004.05

В.О. ГОРБАТЮК, аспірант, Інститут кібернетики імені В.М. Глушкова НАН України,
03187, м. Київ, просп. Академіка Глушкова, 40, Україна,
ORCID: https://orcid.org/0000-0001-7544-0260,
viktor.gorbatiuk@gmail.com

С.О. ГОРБАТЮК, молодший науковий співробітник, відділ теорії цифрових автоматів,
Інститут кібернетики імені В.М. Глушкова НАН України,
03187, м. Київ, просп. Академіка Глушкова, 40, Україна,
ORCID: https://orcid.org/0000-0001-6834-4211,
gorbatiuk_sergiy@i.ua

МЕТОДИ ПЕРЕВІРКИ АЛГЕБРАЇЧНИМ СПІВСТАВЛЕННЯМ СПРОТИВУ HTTP-АТАКАМ НА РОЗУМНИЙ БУДИНОК

Ключову роль для створення безпечного середовища грає якнайшвидше виявлення вторгнень та можливість знешкодити або ізолювати порушника при кібератаці. Дана робота має мету показати поширені види кібератак на розумні будинки, а також інструменти та методи для їх виявлення. Зокрема в роботі розглянуто метод математичного співставлення, що дозволяє на етапі проєктування систем виявляти можливі вразливості й в результаті створювати стабільні веб додатки та сервіси, а на етапі експлуатації оцінювати імовірність атаки на систему та прогнозувати наслідки.

Ключові слова: *кібербезпека, HTTP-протокол, кібератака, розумний будинок, спротив атакам, алгебраїчне моделювання, алгебраїчне співставлення, формалізація, властивості безпеки.*

Розумний будинок та кібератаки

Розумний будинок — це система датчиків і технічних рішень, об'єднаних в єдину систему і здатних виконувати певні задані дії та вирішувати певні повсякденні завдання без втручання людини. Інтернет речей (*IoT*) — це механізм, який наразі забезпечує роботу розумних будинків. На сьогоднішній день *HTTP*-протокол є найчастішим рішенням для забезпечення роботи *IoT*. Усі програмовані пристрої, «розумні» прилади та механізми в сучасних розумних будинках підключено до мережі Інтернет.

Кібератака — це напад, здійснений кіберзлочинцями за допомогою одного чи кількох комп'ютерів на один чи декілька комп'ютерів чи мереж. Кібератака може вимкнути комп'ютери зі зловмисною метою, викрасти дані або використовувати атакований комп'ютер як точку запуску для інших атак.

Без програм кіберзахисту розумний будинок не зможе захистити себе від спроб вторгнення та стане доступною мішенню для кіберзлочинців. Ризики атак зростають через безперервне підключення та використання хмарних сервісів, таких як *Amazon Web Services*, для зберігання конфіденційних даних та особистої інформації. Нині є вельми поширеною невідповідна конфігурація хмарних сервісів, що у поєднанні з дедалі орга-

нізованішими та складнішими кіберзлочинами означає, що зростає ризик успішної кібератаки або викрадення даних розумного будинку. Компанії, що надають послуги сервісів для розумних будинків, уже не можуть покладатися лише на готові рішення кіберзахисту, такі як антивірусне програмне забезпечення та мережеві екрани. Тактика кіберзлочинців стає дієвішою та агресивнішою щодо звичайних засобів кіберзахисту. Зростання кількості випадків порушень безпеки даних означає, що підтримка кібербезпеки є необхідною не лише для жорстко регульованих галузей промисловості, бо навіть малі підприємства підпадають під ризики зазнати непоправної шкоди репутації після злому та викрадення даних.

Згідно з аналізом інцидентів мережевої безпеки, 75% атак були націлені на веб-додатки. Однак ці атаки не можна легко виявити чи відбити. Більшість проблем сталося через нехтування програмної безпеки під час розробки додатків або платформ. Розробники додатків мають широкий спектр можливостей, але на етапі розробки зосереджують увагу над основним функціоналом і часто не враховують вимоги до безпеки. Як наслідок, нерідко створюються додатки, небезпечні під кутом зору інформаційної безпеки. Ці фактори призводять не лише до уразливості захисту системи або веб-додатків, а й до логічного дефекту або уразливості потоку операцій, а також спричиняють втрати майна та репутації самого підприємства через ці проблеми управління.

По суті, наше суспільство є технологічно залежнішим, ніж будь-коли раніше, і немає жодних ознак того, що ця тенденція сповільниться. Саме тому значення кібербезпеки зростає. Витік даних у системах розумного дому, що мають високий рівень інтеграції із соціальними мережами, можуть призвести до викрадення конфіденційної інформації, такої як дані кредитної картки та дані банківського рахунку, що тепер зберігаються у хмарних службах зберігання *Dropbox* або *Google Drive*.

Незалежно від того, хто є користувачем: приватна особа, малий бізнес або велика транснаціональна компанія, ці користувачі щодня покладаються на комп'ютерні системи. У поєднані зі зростанням ролі та популярності хмарних середовищ і недостатнім захистом хмарних сервісів, смартфонів та Інтернету речей (*IoT*), ми маємо безліч потенційних вразливостей безпеки, яких не було кілька десятиліть тому.

Різновиди атак в розумних будинках

Незалежно від того, чи є в будинку повноцінна комплексна система керування для розумного будинку чи просто набір розумних пристроїв, ми повинні оцінювати безпеку як сукупність безпеки кожного приладу. Згідно з міжнародними дослідженнями та статистикою компанії *Avast*, що надає послуги в сфері кіберзахисту та є одним із найбільших розробників антивірусного програмного забезпечення, 40,8% розумних будинків мають принаймні один пристрій, уразливий до загроз кібербезпеки. Водночас 31,4% будинків перебувають у зоні ризику через невиправлені вразливості програмного забезпечення. Єдиний спосіб захиститися від потенційної загрози – це уважніше ставитися до пристроїв розумного дому, які встановлюються, атаки на такі прилади мало чим відрізняються від загальнопоширених мережевих атак.

Методи мережевих атак класифікують як "пасивні" та "активні". Пасивні атаки – це перехват інформації на шляху до отримувача без втручання. Активні атаки – це мережеві атаки, під час яких хакер або зловмисник намагається внести зміни в дані на цільовому об'єкті або дані на шляху до цілі. Такі активні атаки поділяються, своєю чергою, на "підробку", "зміну повідомлення" та "відмову в роботі".

Усі пасивні атаки ми можемо, по суті, назвати *розвідувальними атаками*. Тобто розвідувальні атаки – це атаки для збору загальної інформації. Шпигунство (*Snooping*, також відоме як "підглядання" або "збір інформації") –

це використання доступу до приватної інформації. Ця інформація може бути використана з вигодою, наприклад, для отримання комерційних секретів компанії, які допоможуть у власному бізнесі або в прийнятті рішень щодо купівлі акцій. Шпигунство також можна використовувати для активних нападів, таких як шантаж. Розвідувальні атаки можуть здійснюватися і через логічний, і фізичний підхід, тобто інформація може збиратися за допомогою сканування мережі або через соціальну інженерію та фізичне стеження. Пінгування – це тестове опитування утилітою *Ping* з метою встановлення відповідності між *IP*-адресою та доменним іменем. Деякі поширені приклади розвідувальних атак це: вишукування пакетів, розгортання пінгування, сканування портів, зловмисна розсилка, соціальна інженерія та інформаційні запити в Інтернеті.

Ми можемо розглянути розвідувальні атаки, розділивши їх на логічні та фізичні [1].

Логічна розвідка включає збір інформації про все, що відбувається в цифровому світі без безпосереднього фізичного контакту з людиною. Наприклад, пінгування та сканування портів – це два методи виявлення чи підключена система і що вона шукає в мережі. Відповідь при скануванні портів може бути виявленням того, чи прослуховує *IP*-адреса порт 443 для потоку даних *HTTPS*-протоколу. Це дає хакеру змогу дізнатися, чи може він використовувати *HTTPS* для своїх цілей.

Ще одним прикладом логічних розвідувальних атак є атаки мережі "людина-посередник" (*MITM*), які відбуваються, коли зловмисні сторони перехоплюють потік даних, що передається між мережами та зовнішніми джерелами даних або всередині мережі. У більшості випадків хакери досягають успішних атак "посередник" за допомогою слабких протоколів безпеки. Вони дають хакерам змогу передавати себе як ретрансляційний або проміжний обліковий запис і маніпулювати даними під час транзакцій у реальному часі.

Крім того, ми бачимо інформаційні запити через Інтернет, які іноді називають запитами *whois*. Доменні імена – ієрархічні імена мережі Інтернет, що обслуговується групою серверів та централізовано ними адмініструються. Проблема схожа на патентування назви продукту або бренду, коли компанія «*A*» хоче використовувати зазначене доменне ім'я, а компанія «*B*» вже володіє цим доменним ім'ям. Усі доменні імена зареєстровані на незалежні компанії та належать цим компаніям, оскільки має відбуватися регулювання та обслуговування цих доменних імен. Ці платформи управління (вони ж власники) доменними іменами виконують обмін їх, функціонування та підтримку від створення до закінчення терміну дії. Ці послуги обслуговування доменного імені зазвичай пропонують багато інформації про організацію, включно з контактними даними та контактною інформацією. Усе це значно полегшує збір інформації, коли ви звертаєтеся до компанії, яка має законну інформацію про цікаву вам особу.

Фізична розвідка виходить за межі того, що контролює мережевий адміністратор. Є елементи, які ніколи не будуть вповні захищені, наприклад, фізичне розташування, а також елементи безпеки, такі як камери, дверні замки або охоронці. Однак це може вплинути на фізичну безпеку мережі.

Атаки доступу (активні атаки) вимагають наявності певної можливості вторгнення. Це може бути щось просте, як отримання облікових даних власника облікового запису для підключення обладнання безпосередньо до мережевої інфраструктури. Атаки доступу, так само як і пасивні розвідувальні атаки, можна поділити на логічні та фізичні, при цьому логічні здійснюються через мережу, а фізичні радше схиляються до соціальної інженерії.

Логічні атаки доступу, такі як атаки грубої сили або перевірка паролів у мережі за допомогою таблиць або словників, як правило, створюють великий потік даних у мережі, і їх може легко помітити навіть недосвідчений оператор моніторингу мережі. Саме тому більшість атак логічного доступу зазвичай здійснюються після того, як було

отримано достатньо даних або повноважень. Є також тенденція попередньо вдаватися до пасивного нападу, наприклад, атака «людини-посередника», щоб спробувати зібрати більше інформації перед активною атакою доступу.

Неперевірені дані користувача можуть поставити організаційні мережі під загрозу так званої атаки *SQL*-ін'єкцій, тобто коли відбувається введення зловмисницького коду *SQL*. За методом мережевої атаки зовнішні сторони маніпулюють формами, надсилаючи шкідливі коди замість очікуваних значень даних. Вони ставлять під загрозу мережу та отримують доступ до конфіденційних даних, таких як паролі користувачів [2].

У результаті атаки *Ransomware* зловмисники шифрують канали доступу до даних, утримуючи ключі дешифрування – модель, яка дає хакерам змогу вимагати з постраждалих викуп за ці ключі. Платіжні канали зазвичай мають рахунки криптовалюти, які не можна відстежити. Хоча органи кібербезпеки не рекомендують платити зловмисникам, деякі організації обирають цей шлях як швидке рішення для відновлення доступу до даних.

Маскування – це дія, коли мережевий пристрій під час атаки видає себе за інший пристрій. Цей підхід є ідеальним, якщо зловмисник хоче залишитися непоміченим. Якщо пристрій може успішно обдурити цільову мережу, яка після перевірки визначає його, як авторизований пристрій, зловмисник отримує всі права доступу, які авторизований пристрій встановив під час входу. Крім того, не буде жодних попереджень щодо безпеки. Навіть модератор, який сканує записи потоку даних, не побачить нічого поганого, якщо зловмисник не зробить того, що звичайний користувач не зробив би, наприклад, спробує отримати доступ до системних областей. У маскуванні також можливий фізичний підхід, який може бути так само ефективним. Якщо ви залишите свій пристрій зареєстрованим і відійдете, зловмисник може сісти за пристрій й отримати ваші права доступу. Це є той же самий принцип.

Фізичний доступ – це доступ до обладнання або доступ до людей. Соціальна інженерія є дуже небезпечною й від неї важко захиститися просто тому, що користувачі мережі зазвичай є найслабшою ланкою кібербезпеки. Найпростіший тип атаки соціальної інженерії – це розсилка зловмисних електронних листів, призначених для того, щоб знайти цільову аудиторію, або встановлення програми для запису облікових даних на комп'ютер людини, яка має доступ. Навіть спеціалісти з кібербезпеки можуть бути уразливими щодо таких атак просто тому, що вони люди, а люди роблять помилки.

Атака «Відмова в роботі» (*DoS*) дуже відрізняється від інших категорій атак і за технікою виконання, і за цілями. У той час, як інші надають зловмиснику додаткові привілеї, *DoS*-атака зазвичай блокує всіх, включно і зловмисника. Мета атаки *DoS* – завдати шкоди цілі, унеможливлюючи роботу мережі. Відмова в роботі означає, що мережа не може приймати будь-який потік даних. Це може статися через збій електроживлення або затоплення мережі непотрібним потоком даних, що заважає функціонуванню мережі. І те, й інше історично відбувалося без будь-яких шкідливих намірів, і в обидвох випадках відмові в роботі можна запобігти за допомогою фізичних і логічних блокаторів.

Огляд програмних платформ, що виявляють атаки

У наступному розділі здійснено аналіз нинішнього стану технологій для подолання проблем захисту інформації та розглянуто застосунки, що використовуються для запобігання атакам і для виявлення вразливостей, пов'язаних із кібербезпекою в розумних будинках.

Платформи для виявлення *DdoS*-атак

Fastnetmon — це програма з відкритим вихідним кодом, який пропонує сервіс, що працює на сервері *Linux* [3]. Це дуже високо-

ефективний детектор *DDoS*, побудований на основі кількох механізмів захоплення пакетів [4]. Він підтримує ряд механізмів захоплення, таких як дзеркальне відображення портів, *NetFlow, sFLOW, IPFIX* тощо, щоб збирати інформацію про вхідний потік даних. Програма може виявити атаку на певні *IP*-адреси в мережі на основі пропускної здатнос-ті, кількості пакетів за секунду або кількості потоків. Можна визначити та налаштувати ці параметри на основі профілю атаки. Наступна дія – повідомити маршрутизатору про відкидання шкідливого потоку даних та про відповідні правила *BGP blackhole* або *BGP flow*, щоб позначити цей конкретний маршрут. *Fastnetmon* пропонує параметри, за допомогою яких можна визначити, як довго *IP*-адреса залишається заблокованою і коли її можна знову дозволити. Він може бути інтегрований у будь-яку існуючу мережу без будь-яких змін і додаткового обладнання.

Платформа під назвою ***HADEC*** призначена для виявлення діючих високошвидкісних *DDoS*-атак, які відбуваються на мережевих і прикладних рівнях, таких як *TCP-SYN, HTTP GET, UDP* і *ICMP* [5]. Структура складається з двох основних компонентів: сервера виявлення та сервера захоплення. Виявлення *DDoS* у реальному часі починається із сервера захоплення, який відповідає за захоплення реального мережевого потоку даних, і передачі журналу на сервер виявлення для обробки. Виявлення обчислює вхідний пакет для *UDP*, *ICMP* і *HTTP*, щоб виявити атаку, якщо вихідне з'єднання перевищує попередньо визначений поріг. Запропонований детектор забезпечує недорогі рішення для фінансових установ, а також малих і середніх компаній [6].

Метод виявлення під назвою *D-FACE* використовується для виявлення чотирьох типів потоку даних: легального користувача, низького, високошвидкісного та флеш-трафіку [7]. Флеш-трафік — це різке зростання або сплеск трафіку на певний веб-ресурс, що призводить до його відмови в роботі. Виявлен-ня використовує різницю ентропії, яка містить нормальний потік даних, тоді як значення ентропії *IP* джерела є матрицею виявлення для розрахунку атаки. Виявлення починається з вилучення відповідного заголовка, який класифікує мережу в унікальний мережевий потік. Розділення низького, високого потоку даних і трафіку флеш-подій базується на порівнянні поточної швидкості вхідного потоку даних у кожному часовому вікні та на основі значення інформаційного потоку даних.

Існує також метод, який виявляє *HTTP DDoS*-атаку за допомогою машинного навчання щоб відрізнити ботнет від легальних користувачів у виявленні потоку даних атаки, автентичного потоку даних та флеш-трафіку [8]. Запропонована система розгорнута як проксі-сервер та виконує перевірку поведінки користувачів замість моніторингу всього потоку даних. Запропонована робота виявляє джерело ботнету та досліджує поведінку користувачів для виявлення шкідливого запиту до веб-сервера.

Матриця машинного навчання для визначення часових інтервалів, що базується на біологічному алгоритмі ехолокації кажанів, дає змогу швидко та завчасно виявляти *HTTP DDoS*-атаки [9]. Робота використовувала часові інтервали замість сеансів користувача та шаблони пакетів для створення алгоритму виявлення. Часовий інтервал використовує матрицю машинного навчання, призначаючи значення максимальної кількості сеансів для одноразового інтервалу та обчислюючи кількість сеансів за одноразовий інтервал для виявлення *DDoS*-атаки на прикладних рівнях. Матриця також враховує дві сторінки запиту *HTTP GET*. Частота доступу користувачів до веб-сторінки та часовий проміжок між запитом першої сторінки та другою сторінкою визначаються для моніторингу поведінки користувачів.

Ще однією програмою є хмарне виявлення *HTTP DDoS*-атак за допомогою статистичного підходу з коваріаційною матрицею [10]. Виявлення запровадило два алгоритми, відомі як навчання та тестування, щоб розпізнати різні типи *HTTP*-атак на основі поведінки того, хто атакує. Алгоритм навчання вико-

ристовувався для побудови звичайних шаблонів мережевого потоку даних, а алгоритм тестування використовувався для визначення типів отриманого потоку даних. Результати, отримані в результаті цього дослідження, були оцінені за допомогою матриці плутанини для вимірювання ефективності виявлення та надання результатів внутрішнього та зовнішнього хмарного середовища. Матриця плутанини є ресурсом візуалізації даних. Матриця плутанини – це спосіб побудови таблиці, що дає змогу відразу бачити результати використання алгоритму. Використання простої таблиці для показу аналітичних результатів значно спрощує процес прогнозування.

Платформи виявлення вторгнень на базі сервера

Snort — це система запобігання та виявлення вторгнень у мережу з відкритим кодом (*IDS/IPS*), розроблена *Sourcefire*. *Snort* може виконувати аналіз протоколів і пошук/відповідність вмісту. Його можна використовувати для виявлення різноманітних атак і втручань, таких як переповнення буфера, приховане сканування портів, *CGI*-атаки, *SMB*-проби, спроби ідентифікації ОС та багато іншого. *Snort* також має можливість сповіщення в реальному часі, включно з механізмами попередження для системного журналу, файлу, визначеного користувачем, сокету *UNIX* або повідомлень *WinPopup* для клієнтів *Windows*. *Snort* має три основні види використання: прямий аналізатор пакетів, такий як *tcpdump*, реєстратор пакетів (корисний для налагодження мережевого потоку даних тощо) або повноцінну систему запобігання вторгненню в мережу [11].

Fail2Ban сканує файли журналів, такі як */var/log/auth.log*, і забороняє *IP*-адреси, які мають занадто багато невдалих спроб входу. Це робиться шляхом оновлення правил системного мережевого екрану, щоб відхиляти нові з'єднання з цих *IP*-адрес протягом певного періоду часу. *Fail2Ban* готовий до зчитування багатьох стандартних файлів журналів, наприклад, для *sshd i Apache*, і його легко налаштувати на читання будь-якого файлу журналу, який ви оберете, для будь-якої вибраної помилки. Хоча *Fail2Ban* здатен знизити частоту неправильних спроб автентифікації, він не може усунути ризик, який представляє слабка автентифікація. Необхідно налаштувати служби на використання лише двох факторів або механізмів публічної/приватної автентифікації для захисту служб [12].

FuzzDB було створено, щоб підвищити інтерес до ймовірності виникнення та виявлення умов безпеки за допомогою динамічного тестування безпеки програми. Це перший і найповніший відкритий словник шаблонів зловмисницьких ін'єкцій, передбачуваного розташування ресурсів і регулярних виразів для відповідності відповідей сервера. Шаблони атак – *FuzzDB* містять вичерпні списки примітивів загрузочних даних атак для тестування шкідливих ін'єкцій. Ці шаблони, класифіковані за типом атак та, якщо доречно, за відомим типом платформи, у яких виникають такі проблеми, наприклад ін'єкція команд ОС, списки каталогів, обхід каталогів, доступ до джерела, обхід завантаження файлів, обхід автентифікації, *XSS*, ін'єкції *crlf* заголовка *http*, ін'єкція *SQL*, ін'єкція *NoSQL*, і інші. Ін'єкції нульового байта — це техніка використання вразливості, яка використовується для обходу фільтрів у веб-інфраструктурі шляхом додавання нульових байтових символів у *URL*-адресі (тобто %00 або 0x00 у шістнадцятковому) до наданих користувачем даних. *FuzzDB* каталогізує 56 шаблонів, які потенційно можна інтерпретувати як нульовий байт, і містить списки часто використовуваних методів «*get, put, test*» та пари «ключ – значення», що ініціюють режим налагодження [13].

Для виявлення та запобігання використанню відомих і поширених вразливостей організація *OWASP* визначила загальний набір правил, відомий як основний набір правил *OWASP* [14] (*OWASP CRS*). *OWASP CRS* широко використовується у великих організаціях, таких як *Akamai, Azure, CloudFare, Fastly i Verizon*. Завданням *OWASP CRS* є надання набору

загальних правил виявлення атак, які при передачі до мережевого екрану веб-додатків *MODSECURITY* забезпечують базовий рівень захисту для будь-якого веб-додатка. *OWASP CRS* реалізує негативну модель, де правила розроблено для виявлення відомих моделей атак [15].

Мережеві засоби виявлення вторгнень

Системи виявлення вторгнень на основі мережі виявляють вторгнення, перехоплюючи пакети з мережі та застосовуючи набір сигнатур або шаблонів атак. Приклади цього сімейства інструментів включають *Network Flight Recorder* [16], *Bro* [17], *RealSecure* [18] або *NetRanger* [19].

Застосування алгебраїчного методу

Основні поширені методи та інструменти, які використовувалися для захисту традиційних інформаційних технологій від кібератак, з часом втрачають здатність цілковито запобігти успішному проникненню шкідливих програм у систему. Тому потрібні нові підходи. Хоча системи й захищено засобами *IT*-безпеки, зловмисники все ж знаходять спосіб отримати несанкціонований доступ і скомпрометувати їх за допомогою кібератак. Ці кібератаки необхідно виявити з прийнятною частотою хибних тривог якомога швидше, а також ідентифікувати й ізолювати. Отже, є нагальна потреба в ефективній системі оцінки здатності спротиву системи відомим атакам з використанням ефективних математичних методів, зокрема, формальних методів.

В Інституті кібернетики ім. В.М. Глушкова використовується технологія інсерційного моделювання для виявлення вразливостей у програмних системах і теорія взаємодій агентів та середовищ. Розглядаються моделі взаємодій програмних систем у рамках мережевого середовища, у якому може відбуватися зловмисна діяльність або атака.

Для формального опису моделі використовують специфікації алгебри поведінок, а за основу для формальних методів верифікації беруться методи символьного моделювання та автоматичного доведення теорем. У системах із довільною кількістю агентів алгебраїчний підхід та інсерційне моделювання дають нам змогу довести або спростувати властивості такої системи. Ми можемо генерувати різні сценарії взаємодії агентів або груп агентів, використовуючи абстрактну формальну модель програми. Такі сценарії мають символьну форму й ілюструються контрприкладами. Згенеровані символьні сценарії дають повне уявлення про поведінку, а в результаті можуть використовуватися для тестування на етапі створеного програмного забезпечення.

Для складних розподілених систем із безліччю агентів для побудови моделей та моделювання взаємодії агентів із середовищем інсерційне моделювання є одним із ефективних методів. Основна концепція інерційного моделювання – це створення чіткої ієрархії від середовища до агентів, що входять у ці середовища, взаємодія агентів із середовищами різних рівнів, їх взаємний вплив одне на одного, а також зміни поведінки групи агентів, коли середовище змінюється. Середовище може виступати як агент, який також може бути занурений в інше середовище. У таких системах стани визначаються значеннями атрибутів, а агенти розглядаються як системи переходу атрибутів. Агенти описано за допомогою набору атрибутів, які й визначають тип агента, а атрибути середовища пов'язані з глобальними атрибутами, які відомі всім агентам.

Алгебра поведінок є двохсортною алгеброю над множиною поведінок та дій агентів. Поведінка описується за допомогою поведінкових рівнянь, що складаються з поведінкових виразів, які, своєю чергою, містять операції – “.” (префіксінг), “+” (недетермінований вибір), “;” (послідовна композиція поведінок), “||” (паралельна композиція). Дії агентів визначаються за допомогою перед- та післяумови в термінах відповідної теорії й ілюструються

процесною компонентою. Приклад формалізації протоколу й атаки показано далі.

Формалізація протоколу *HTPP* та простої атаки

Ми розглядаємо протокол *HTTP*, як взаємодію агентів в мережевому середовищ. Кожен агента має *IP*-ім'я та визначається перелічувальним типом *IP_NAME*, що містить всі можливі *IP*-імена. Значення атрибута є символьним рядком, наприклад: 192.168.1.1. Відповідно кожен агент має мережеву адресу, що також визначається множиною перелічувального типу *MAC_NAME*. Значення атрибута є також символьним, наприклад: 00:00:5e:00:53:af.

Ми розглядаємо тип агента *NODE*, який визначається своїми атрибутами, а саме:

- *IP:IP_NAME – IP* адреса агента.
- *list_IP: (int) -> IP_NAME* – функціональний атрибут агента, що містить IP-агентів з таблиці в якій записані адреси всіх агентів, яким колись було надіслано повідомлення, або отримано.
- M – кількість рядків у таблиці, або кількість адрес з якими контактував агент.
- *MAC:MAC_NAME – MAC*-адреса агента.
- *list_MAC: (int) -> MAC_NAME – MAC*-адреси всіх агентів в таблиці.

Кожен агент типу *NODE* має ім'я – a_1, a_2,.... Агент середовища може бути визначений і чесним, і злочинцем. При взаємодії агенти виконують дії, що відповідають протоко-лу обміну повідомленнями. Такими є наступні дії, що параметризовані відповідними значеннями арибутів.

Дія *SendRequest* (*x, z*) відправляє повідомлення *Request*, де *x* – агент відправник, *z* – *IP*-отримувача, що має відповідну *MAC*-адресу. Запис відправляється лише тим отримувачам, які є в списку, тобто є передумова дії, повідомлення *Request*(*x.IP, u*) відправляється за *MAC*-адресою *u* якщо існує такий *IP* в таблиці відправника.

SendRequest(*x, z*) = (*Exist i:int*)(*z* == *x.list_IP*(*i*) && (1 <= *i* <= *x.M*)) -> "*send Request*(*x.IP, list_MAC*(*i*))" 1

GetRequest, агент отримує запит *Request*(*y, u*), де *y* – *IP*-агента відправника, *u* – *MAC* адреса отримувача. Ця дія також має передумову, що *Request* отримує лише той агент, *MAC*-адреса якого співпадає із другим параметром сповіщення. В цій же дії відповідний агент надсилає відповідь на запит – *Response* за *MAC*-адресою, яку він знайшов у своєму списку згідно з *IP*-відправника.

GetRequest = (*Exist x:NODE, y:IP_NAME, u:MAC_NAME,i:int*) (*y* == *x.list_IP*(*i*)) &&

(*1* <= *i* <= *x.M*) && (*u* == *x.MAC*) -> "*receive Request* (*y,u*), *send Response*(*x.IP, x.list_MAC*(*i*))"1

Аналогічно визначаємо протокол для відправника, що отримує відповідь на запит, а саме сповіщення *Response*.

GetResponse = (*Exist x:NODE, u:MAC_NAME, y:IP_NAME*) (*u* == *x.MAC*) -> "*receive Response* (*y, u*)" 1

Дія *NoSendRequest(x,z)* – є дія, в якій *z* не знаходиться в списку адрес агента *x* і сповіщення не відправляється.

NoSendRequest(*x, z*) = (*Forall i:int*)(*z* != *x.list_IP*(*i*) && (*1* <= *i* <= *x.M*))-> "" 1

У випадку, якщо адреса не міститься у списку агента, він надсилає запит до всіх агентів у мережі, щоб ідентифікувати потрібного, та надсилає свою адресу

SendARPRequest (*x,z*) = (*Forall y:NODE*) -> "*send Broadcast*(*x.IP, x.MAC, z*)" 1

Отримання *Broadcast* повідомлення агентом *x*, який отримав запит на свою адресу, відбувається за допомогою дії *GetARPRequest.* В цій же дії агент відправляє повідомлення про свою *MAC*-адресу за адресою відправника.

GetARPRequest = (*Exist x:NODE, y:IP_NAME, u:MAC_NAME, z:IP_NAME*) (*z* == *x.IP*) -> "*receive Broadcast*(*y, u, z*), *send ARPResponse* (*x.IP, x.MAC, u*)" 1

Агент, що шукав адресу, отримує *ARPResponse* та вносить її до свого списку.

GetARPResponseExist = *Exist(x:NODE, y:IP_NAME, z:MAC_NAME, u:MAC_NAME, i:int*) (*x.MAC* == *u*) && (*x.list_IP*(*i*) = *y*) && (*1*<=*i*<=*x.M*) -> "*receive ARPRequest(y,z,u)*" (*list_MAC(i)* = *z*)

GetARPResponseNew = Exist(x:NODE, y:IP_NAME, z:MAC_NAME, u:MAC_NAME) (Forall (i:int) (x.list_IP(i) != y) && (1<=i<=x.M)) && (x.MAC == u) ->"receive ARPRequest(y,z,u)" (x.M == x.M+ + 1; x.list_IP(M + 1) = y; list_MAC(M + 1) = z)

Поведінкове рівняння, що представляє даний протокол буде наступною паралельною композицією агентів:

$B0 = B1(a1,a2) \parallel B1(a1,a3) \parallel ... \parallel B1(a2,a1) \parallel (a2,a3) \parallel ...$,

*B*1(*x,z*) = *AgentRequest*(*x,z*).*B*1,

AgentRequest1(*x,z*) = (*SendRequest*(*x, z.IP*).*GetRequest.SendResponse.GetResponse* + *NoSendRequest*(*x,z.IP*).*SendARPRequest*(*x,z.MAC*).*GetARPRequest.SendARPResponse*. (*GetARPResponseNew* + + *GetARPResponseExist*)

В рівнянні не враховано зникнення сигналу та відсутність вузла із *IP*, що запитується.

Агент зловмисник може скористатися можливістю послати неправдиві дані і вдати нібито його *MAC*-адреса підходить під *IP*-ім'я, що ми запитуємо. Це робиться із метою, щоб перехоплювати сповіщення, що направляються цьому агенту.

Таким чином прибираємо передумову в дії *GetARPRequest z == x.IP,*

GetARPRequest = (*Exist x:NODE, y:IP_NAME, u:MAC_NAME, z:IP_NAME*) -> "r*eceive Broadcast*(*y, u, z*), *send ARPResponse* (*x.IP, x.MAC, u*)" 1

Тоді умова того, що агент є зловмисник визначається нерівністю при виконанні протоколу *GetARPRequest*. Таким чином ми можемо записати поведінку зловмисника наступним шаблоном:

X = Z. GetARPRequest, де в шаблоні дії буде записана умова порушення правил (*z* != =*x.IP*) -> "" 1

Ми привели найпростішу формалізацію протоколу для ілюстрації можливості атаки. Весь протокол є поведінковим рівнянням, яке треба вирішити відносно *X* за допомогою символьного моделювання. Таким чином ми визначаємо, що результат атаки досяжний і отримаємо трасу, що веде до цього результату, а саме послідовність відповідних дій. Таким чином ми визначимо, чи є в протоколі запобігання цій атаці чи ні. Щоб запобігти цій атаці треба вставити перевірку. Існує три аномальні прояви атаки, що можуть використовуватися як перевірка:

1. При атаці відповідь надсилається без запиту, тому потрібно перевіряти чи надавався запит

GetARPResponseExist = SendARPRequest -> ...

GetARPResponseNew = SendARPRequest -> ...

2. При атаці не перевіряється чи відправник відправляє свою адресу у такій відповіді, тому потрібно порівнювати надану адресу з адресою відправника

GetARPResponseExist = (*y.MAC == z*) -> ...

GetARPResponseNew = (*y.MAC == z*) -> ...

3. Адресна книга не повинна містити двох однакових адрес

GetARPResponseExist = Forall(*i:int*) (*x.list_MAC*(*i*) != *x.list_MAC*(*i+1*)) -> ...

GetARPResponseNew = Forall(*i:int*) (*x.list_MAC*(*i*) != *x.list_MAC*(*i+1*)) -> ...

Тоді поведінкове рівняння не буде мати рішення.

Виявлення атак алгебраїчним співставленням

Метод алгебраїчного співставлення – це метод виявлення потенційних вразливостей у коді або моделі системи за допомогою порівняння моделі поведінки такої системи із шаблоном атаки. Метод ґрунтується на динамічному аналізі поведінки розв'язанням поведінкових рівнянь.

Модель задається системою рівнянь в алгебрі поведінок, а атака – шаблоном поведінки. При цьому необхідно знайти множину сценаріїв поведінки в заданій системі поведінкових рівнянь, що відповідають даному шаблону або ведуть до нього з початкової поведінки.

Цю задачу можна розділити на дві підзадачі:

1. Знайти послідовність дій, яка відповідає даному шаблону, що зводиться до вирішення поведінкових рівнянь, рішенням яких буде множина сценаріїв поведінок, що відповідають шаблону, або множина сценаріїв поведінок,

що починаються із початкової дії початкової поведінки та ведуть у поведінку шаблону в послідовності інших дій.

2. Доведення досяжності сценарію за допомогою символьного моделювання в тих випадках, коли немає атрибутів, що роблять можливим існування такого сценарію. При такому моделюванні порівнюється середовище моделювання із передумовою дії в шаблоні.

Під час проєктування будь-якої системи рекомендовано й навіть необхідно здійснювати моделювання всіх можливих атак, щоб зрозуміти їхню ймовірність. При спробі зловмисника атакувати мережу, механізм безпеки може розпізнати потенційно небезпечні дії в процесі роботи, але оцінити, за яких умов атака буде успішною, можна лише за допомогою модельної розробки.

Висновки

Ми розглянули більшість відомих на сьогоднішній день атак на розумні будинки, а також інструменти для виявлення цих атак. Велика кількість таких інструментів широко використовується в комерційних масштабах та доволі добре зарекомендували себе, показавши високу ефективність. А проте бувають ситуації, коли заходи безпеки краще впроваджувати на етапі проєктування системи, тому пошук і використання нових підходів залишається актуальним.

Один з таких підходів – це алгебраїчні методи математичного моделювання. Ми застосували метод алгебри поведінок для моделювання атаки «людина-посередник» у мережі розумного будинку й переконалися в можливості його використання для моделювання мережевих атак. Він може бути ефективним і для моделювання атак, і для моделювання мережі як цілого, що дає змогу виявляти проблеми, які навіть не передбачалися.

Надалі планується розглянути та промоделювати інші атаки, щоб довести доцільність методу і його практичну ефек-тивність.

REFERENCES

1. 3 Types of Network Attacks to Watch Out For. [online]. Available at: <https://www.tripwire.com/state-of-security/vulnerability-management/3-types-of-network-attacks/> [Accessed: 23 Sept. 2022].
2. What is a Network Attack? [online]. Available at: <https://www.forcepoint.com/cyber-edu/network-attack/> [Accessed: 23 Apr. 2022].
3. Ultra fast automated DDoS detection & mitigation. [online]. Available at: <https://anuragbhatia.com/2017/10/networking/isp-column/ultra-fast-automated-ddos-detection-mitigation/> [Accessed: 4 May 2022].
4. FastNetmon. [online]. Available at: <https://fastnetmon.com/> [Accessed: 3 Sept. 2022].
5. Hameed, S., Ali, U., 2018. "HADEC: hadoop-based live DDoS detection frame-work", EURASIP Journal on Information Security, vol. 2018, no. 1, p. 11. https://doi.org/10.1186/s13635-018-0081-z.
6. Ghafar, A. Jaafar, Shahidan, M. Abdullah, Saifuladli Ismail, 2019. "Review of Re-cent Detection Methods for HTTP DDoS Attack" Journal of Computer Networks and Communica-tions, vol. 2019, Article ID 1283472, 10 pages, https://doi.org/10.1155/2019/1283472.
7. *Behal, S., Kumar, K., Sachdeva, M.*, 2018. "D-FACE: an anomaly based distributed approach for early detection of DDoS attacks and flash events," Journal of Network and Computer Applications, vol. 111, pp. 49–63.
8. *Singh, K. Singh, P., Kumar, K.*, 2018. "User behavior analytics-based classification of application layer HTTP-GET flood attacks," Journal of Network and Computer Applications, vol. 112, pp. 97–114.
9. *Sreeram, I., Vuppala, V.P.K.*, 2017. "HTTP flood attack detection in application lay-er using machine learning metrics and bio inspired bat algorithm," Applied Computing and Infor-matics, Applied Computing and Informatics, 15(1), DOI:10.1016/j.aci.2017.10.003.
10. *Aborujilah, A. Musa, S.*, 2017. "Cloud-based DDoS HTTP attack detection using covariance matrix approach," Journal of Computer Networks and Communications, vol. 2017, Arti-cle ID 7674594, 8 p.
11. Snort – Network Intrusion Prevention and Detection System. [online]. Available at: <https://www.findbestopensource.com/product/snort> [Accessed: 5 Sept. 2022].

12. Fail2ban - Daemon to ban hosts that cause multiple authentication errors. [online]. Available at: <https://www.findbestopensource.com/product/fail2ban-fail2ban> [Accessed: 23 Apr. 2022].
13. Fuzzdb – Dictionary of attack patterns and primitives for black-box application fault injection and resource discovery. [online]. Available at: <https://www.findbestopensource.com/product/fuzzdb-project-fuzzdb> [Accessed: 23 Apr. 2022].
14. OWASP. Owasp modsecurity core rule set project. [online]. Available at: <https://www.owasp.org/index.php/> [Accessed: 3 Sept. 2022].
15. *Betarte, G., Pardo, A., Martínez, R.*, 2018. "Web Application Attacks Detection Using Machine Learning Techniques," *17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1065-1072, DOI: 10.1109/ICMLA.2018.00174.
16. *Ranum, M.J., Landfield, K., Stolarchuk, M., Sienkiewicz, M., Lambeth, A., Wall, E.*, 1997. "Implementing a generalized tool for network monitoring". *In Proceedings of the Eleventh Systems Administration Conference* (LISA '97) (San Diego, CA).
17. *Paxson, V.,* 1998. "Bro: A system for detecting network intruders in real-time". *In Proceedings of the 7th USENIX Security Symposium* (San Antonio, TX).
18. Internet Security Systems, Inc. RealSecure. 1997. [online]. Available at: <http://www.iss.net/prod/rsds.html> [Accessed: 3 Sept. 2022].
19. Cisco Systems Inc. NetRanger – Enterprise-scale, Real-time, Network Intrusion Detection System. 1998. [online]. Available at: <http://www.cisco.com/warp/public/751/netranger/netra_ds.htm> [Accessed: 3 Sept. 2022].

Received 01.11.2022

V.O. Horbatiuk, postgraduate student, V.M. Glushkov Institute of Cybernetics
of NAS of Ukraine, Glushkov ave., 40, Kyiv, 03187, Ukraine,
ORCID: https://orcid.org/0000-0001-7544-0260,
viktor.gorbatiuk@gmail.com
S.O. Horbatiuk, Junior Researcher, V.M. Glushkov Institute of Cybernetics
of NAS of Ukraine, Glushkov ave., 40, Kyiv, 03187, Ukraine,
ORCID: https://orcid.org/0000-0001-6834-4211,
gorbatiuk_sergiy@i.ua

METHODS FOR CHECKING THE RESISTANCE TO HTTP-ATTACKS ON A SMART HOME BY ALGEBRAIC COMPARISON

Introduction. Cyber-attacks become possible because of vulnerabilities in the IT infrastructure or in a particular system. It is impossible to create the completely secure environment, but it is possible to give sufficient attention to vulnerabilities and reduce the consequences of any attacks that will exploit these vulnerabilities. It is necessary to assume the probability of an attack and be ready to take actions now to prevent them from being successful again. Time is a definite factor in mitigating the damage from a cyber-security breach. Thus, the key role is laid on detecting an intrusion as soon as possible and being able to neutralize or isolate the intruder.

The purpose of this work is to show common types of cyber-attacks on smart homes, as well as detections methods and tools.

Methods. In this way, the method of mathematical comparison works in the work, which allows at the stage of system design to identify the possibilities of vulnerability and, as a result, create stable web applications and services, and at the stage of operation to assess the probability of attacks on the system and predict the consequences.

Results. Algebraic method was applied for "man in middle" attack simulation inside smart home network and it was approved that it can also be used for other network attacks simulation.

Conclusions. It is reasonably to implement security measures during the system design stage so the research and application of new approaches remains actual. Algebraic method may be highly effective for attacks simulations and also for network in general, and it helps to discover problems which would not be predicted.

Keywords: *cyber security, HTTP protocol, cyber-attack, smart home, attack resistance, algebraic modeling, algebraic matching, formalization, security properties.*